Advanced Mac OS X Networking

Session 809

Advanced Mac OS X Networking

Vincent Lubet
Manager, Core OS Networking Team

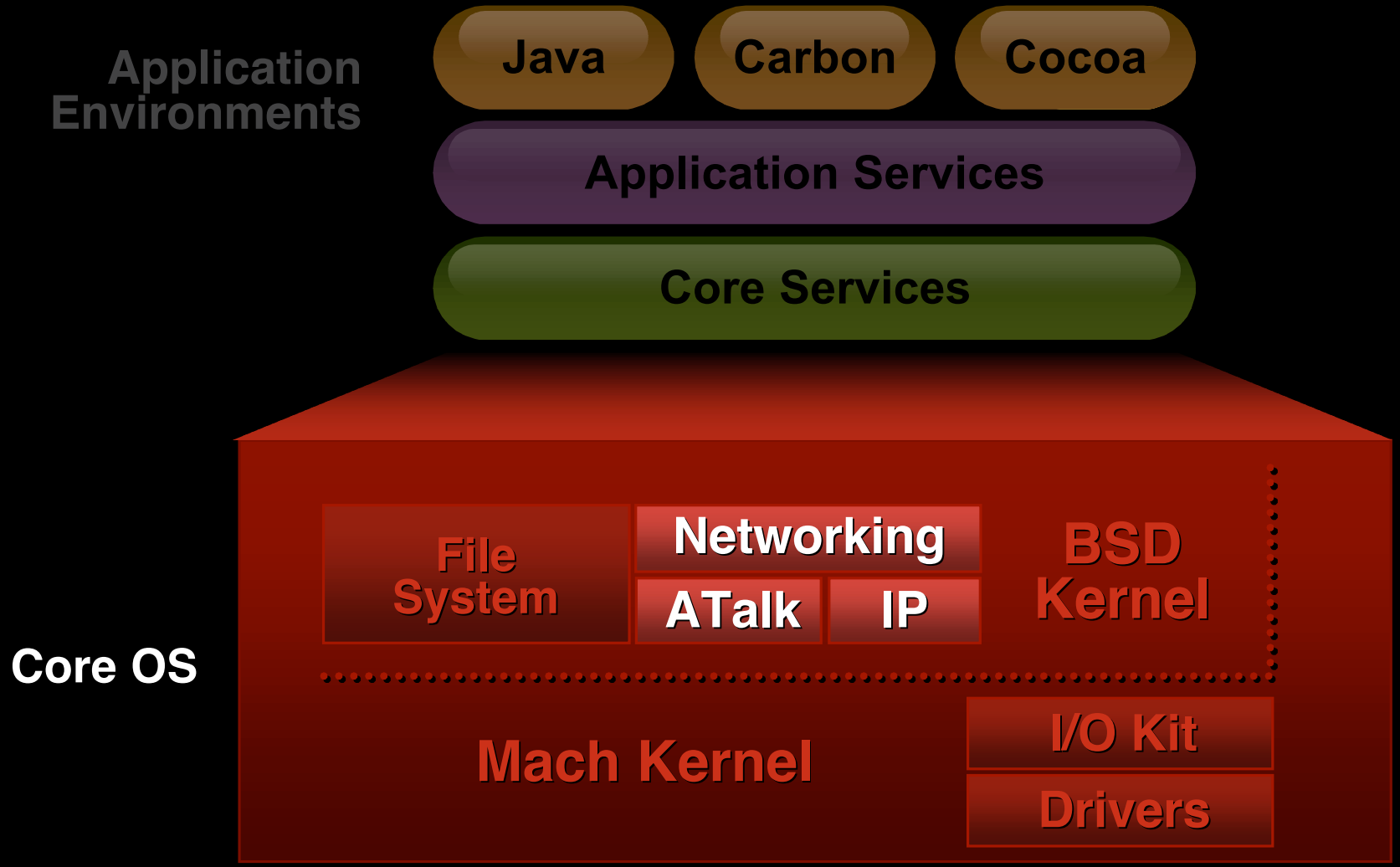
Introduction

- Performance
- NKE
- IPSec
- IPv6

What You Will Learn

- How to take advantage of some of the new networking features
- How to get the best performance for networking applications
- The caveats of kernel programming

Mac OS X Advanced Networking



Performance for Network Applications

Performance

- Do not poll
  - Polling uses 100% of CPU
  - Hurts other processes
  - Wastes energy
  - Shortens portable autonomy
- OT sync-idle events is a form of polling

Performance

- Block or be event driven
- Use blocking mode with threads
  - Recycle threads if you can
- Be event driven
  - For sockets use **select**
  - For OT use async mode with notifiers
  - Run loop for network APIs above Sockets

Performance

- Open Transport on Mac OS X is slow
- A lot context switches
    - Emulation of execution levels
- Lookups are serialized
- 10–15% impact on performance
- Problems compound with number of endpoints

Performance

- Watch for buffer size
- Buffer size is critical for performance
  - Too small = high context switch overhead
  - Too large = starves VM for buffers
- Things to tune
  - Socket buffer size (**SO_SNDBUF, SO_RCVBUF**)
  - Size of buffer passed to send/receive calls
- Size of socket receive buffer directly affects TCP window size

Performance

- Minimize latency
  - Cooperative threads should be avoided
  - Stream data flow, do not wait for all data
- Disk I/O can affect network I/O
  - Same rules apply to both kinds of I/O
- Do not optimize at expense of efficiency
  - Tools: top, fs_usage, sample, . . .
- See Session 906 for more information

Network Kernel Extensions

Network Kernel Extensions

- Extends Networking services
- No need to recompile kernel
- Dynamic loading and unloading
- Running in the kernel comes with responsibilities

NKE Binary Compatibility Issues

- No real API, the Kernel is wide open
- Headers do not discern between what is public and what is private
- Kernel data structures and implementation details are exposed
- We cannot guarantee backward compatibility

NKE Binary Compatibility Issues

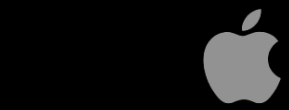
- Kernel data structures and functions will change
  - Bug fixes
  - New functionality
  - Performance enhancements

NKE Alternatives

- Avoid NKE when possible
- Use IOKit for network interfaces
  - IONetworkingFamily for Ethernet
  - IOSerialFamily for PPP
- Use IOKit user client
- Use **PF_NDRV** for protocol handlers in user space

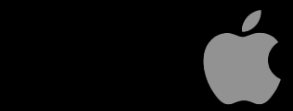
NKE Do and Don't

- Do strip global symbols
- Do not look at kernel data structures
    - Pretend structures are opaque
- Sockets functions OK
    - Still lots of accessors are macros!



Jaguar Impact on NKE

- IPv6 and IPSec added to TCP/IP control block
- **mbuf** routines to isolate implementation
- **soconnect** intercept functions run before protocol
- **soaccept** now returns error instead of a socket
  - Allows for EJUSTRETURN
- NKEMgr deprecated
  - New kernel event protocol in **PF_SYSTEM**
- **ifnet** can be detached



Jaguar Impact on NKE

- We have cleaned up the Dev SDK headers
  - Do not use anything that does not appear in a public header
- We are making private more symbols
- No forward compatibility

NKE Call to Action

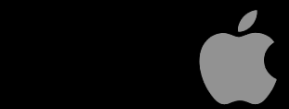
- Plan for sustainable Kernel APIs
- Use nm to list global symbols
- Use kextload to check properties and report common mistakes
- Work with us
    - Seed us with your KEXTs

IPSec

Josh Graessley
Core OS Networking

IPSec: Why

- Secure IP communications
  - Prevent Eavesdropping
  - Authenticate Origin
  - IETF Standard
- Virtual Private Networks
  - IPSec ≠ VPN



IPSec: What

- Per packet Authentication and Encryption
  - Any IP ‘transport’ is supported
  - Transparent to applications
  - Host-to-host, no notion of “users”

IPSec: What

- Authenticate
  - AH (Authentication Header)
- Encrypt
  - ESP (Encapsulated Secure Payload)
- Compress
  - IPComp (IP Compression)

AH: Authentication Header

- Protect IP header and payload
  - Validate originator
  - Detect modifications
  - Detect duplicates
- Supports Multiple Algorithms
  - hmac-md5, hmac-sha1, . . .

ESP: Encapsulated Security Payload

- Encrypt payload
  - Protection against wiretap
  - Protects payload only, not IP header
- Authentication of payload
- Multiple Algorithms
  - Triple DES, AES, blowfish, . . .

IPComp: IP Compression

- Compress payload
    - Compress before encrypt
- Multiple algorithms
    - Jaguar supports inflate/deflate
- In kernel implementation is expensive
- Not widely used

IPSec: Modes

- Tunnel Mode
  - VPN scenario
  - Network-to-network
  - Host-to-network
- Transport Mode
  - Host-to-host

IPSec: Security Policies

- Policy defines filter
  - Source/destination address range
  - Source/destination port range
  - IP protocol (TCP, UDP, ICMP, . . .)
- Policy defines rule
  - Bypass IPSec
  - Require IPSec
  - Use IPsec
  - Discard

IPSec: Security Association (SA)

- Defines algorithm
- Defines key
- One direction only
- May Timeout
    - Based on time
    - Based on traffic

IKE: Internet Key Exchange

- Authenticate remote host
  - Pre-shared key
  - Certificates
  - Kerberos
- Exchange keys
- Negotiate algorithm

IPSec: APIs

- **PF_KEY** socket
  - Set Security Policies
  - Set Security Associations
  - Receive notification of changes
  - Requires root
- **IP_SEC_POLICY** socket option
  - Per socket policy
  - Requires root to bypass IPSec

IPSec: Tools

- setkey
  - Set Security Policies
  - Set Security Associations
- racoon
  - Daemon implementing IKE
  - Exchanges Keys if Security Association is missing
  - Creates Security Association
  - Not running by default

Demo

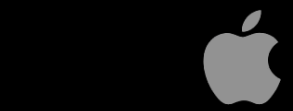
Josh Graessley
Core OS Networking

Advanced Mac OS X Networking: IPv6

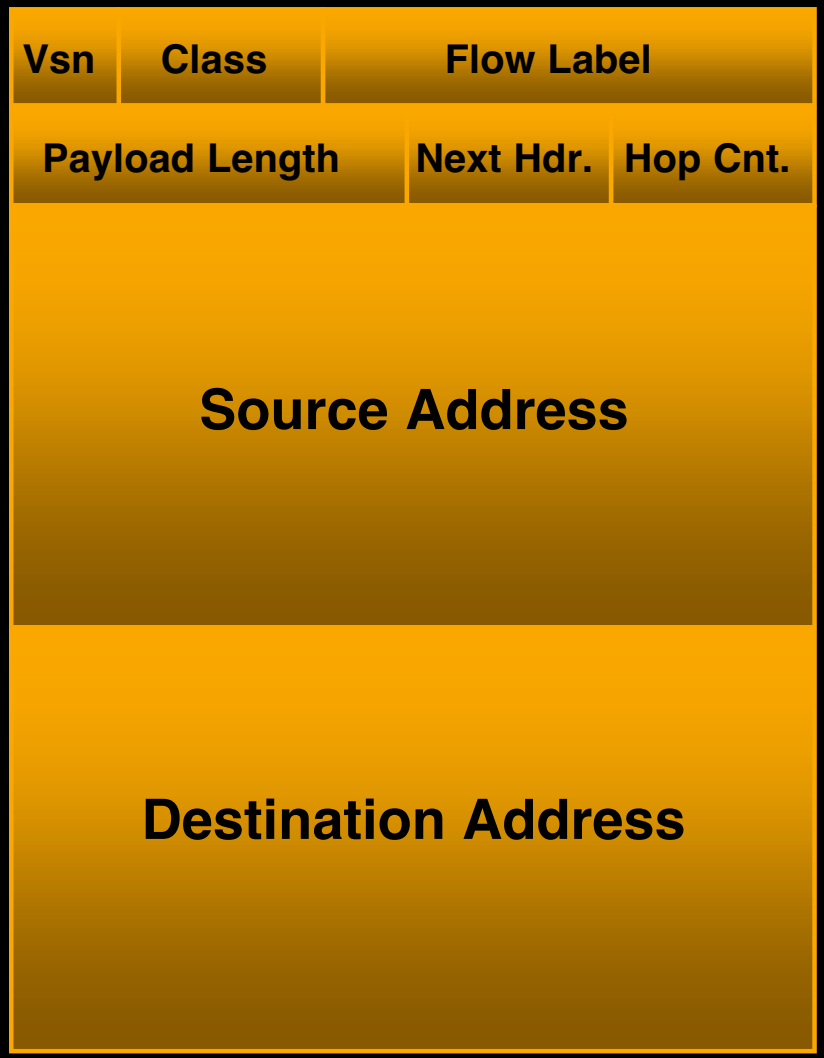
Laurent Dumont
Core OS Networking

Motivations for IPv6

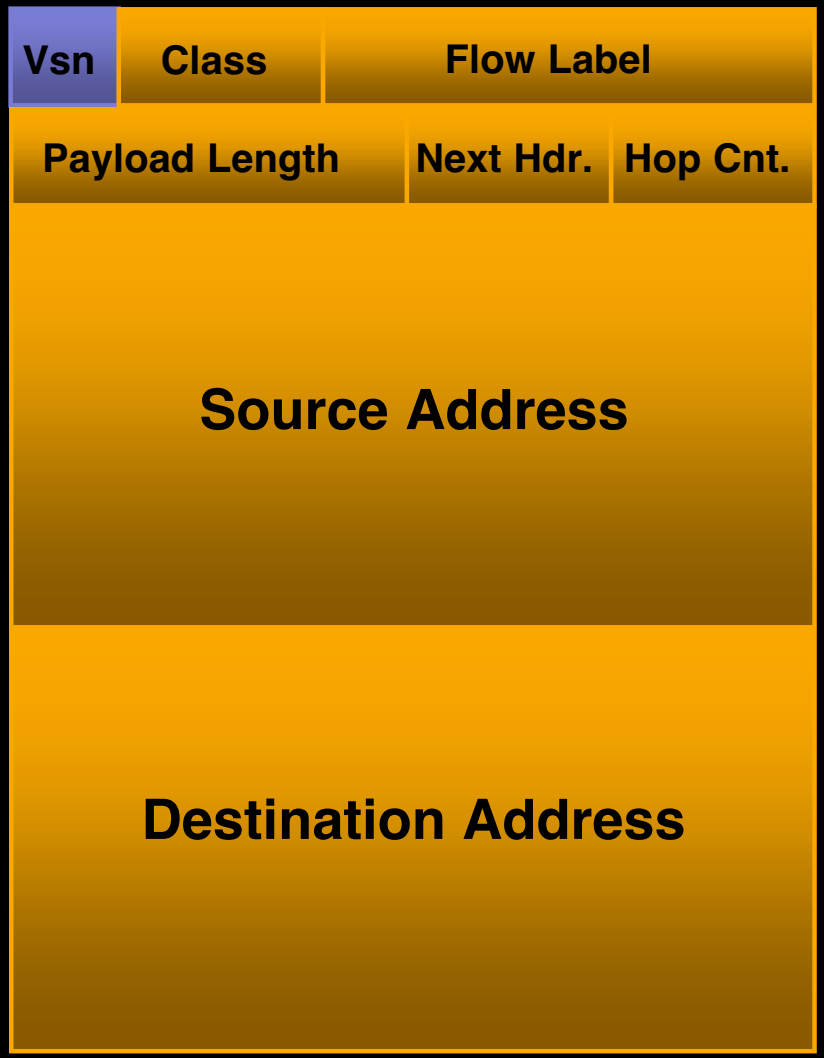
- Addresses Internet growth problems
  - Address Space exhaustion
  - Routing Meltdown
  - NAT breaks end-to-end connectivity
- International markets (Asia, Europe)
  - IPv4 Addresses are scarce
- Includes new standards
  - IPsec, multicast, autoconfiguration



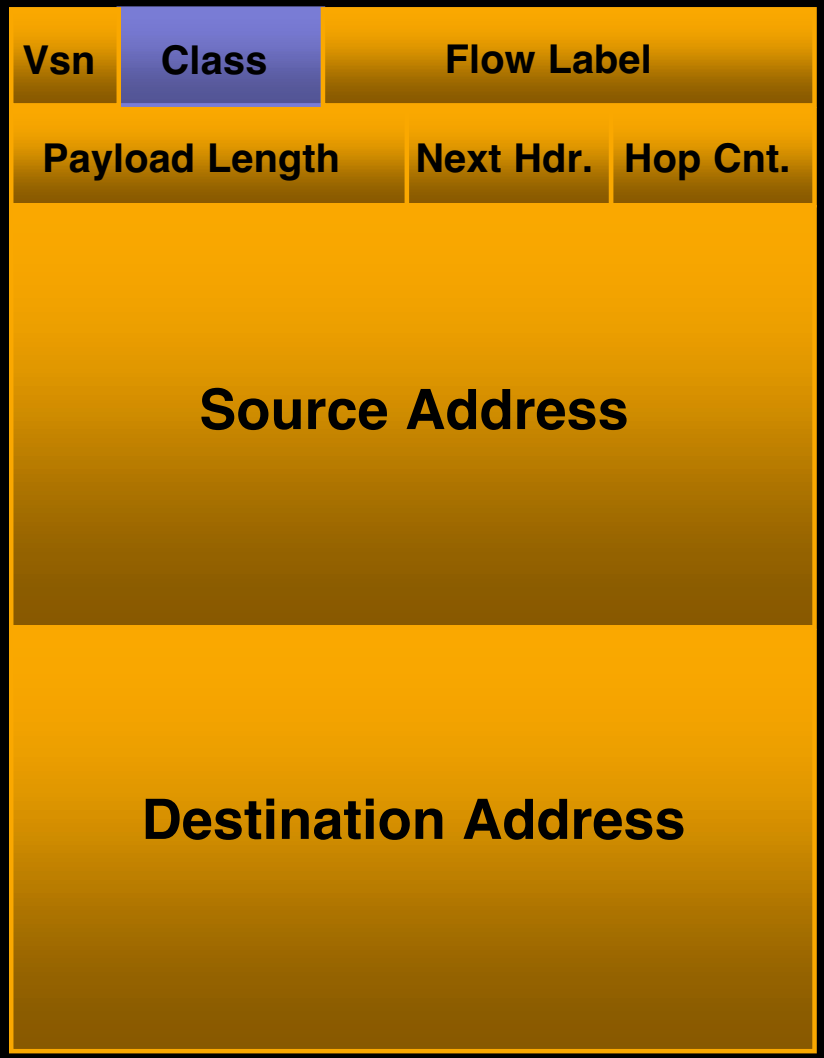
IPv6 Header Structure



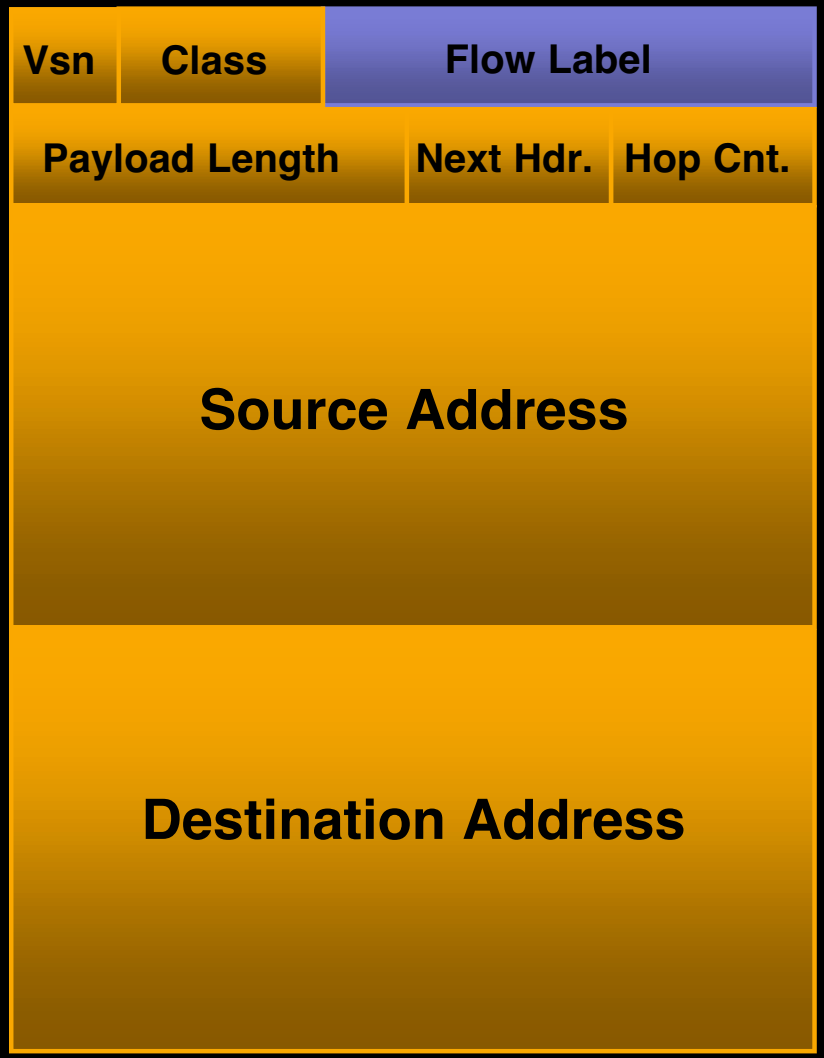
IPv6 Header Structure



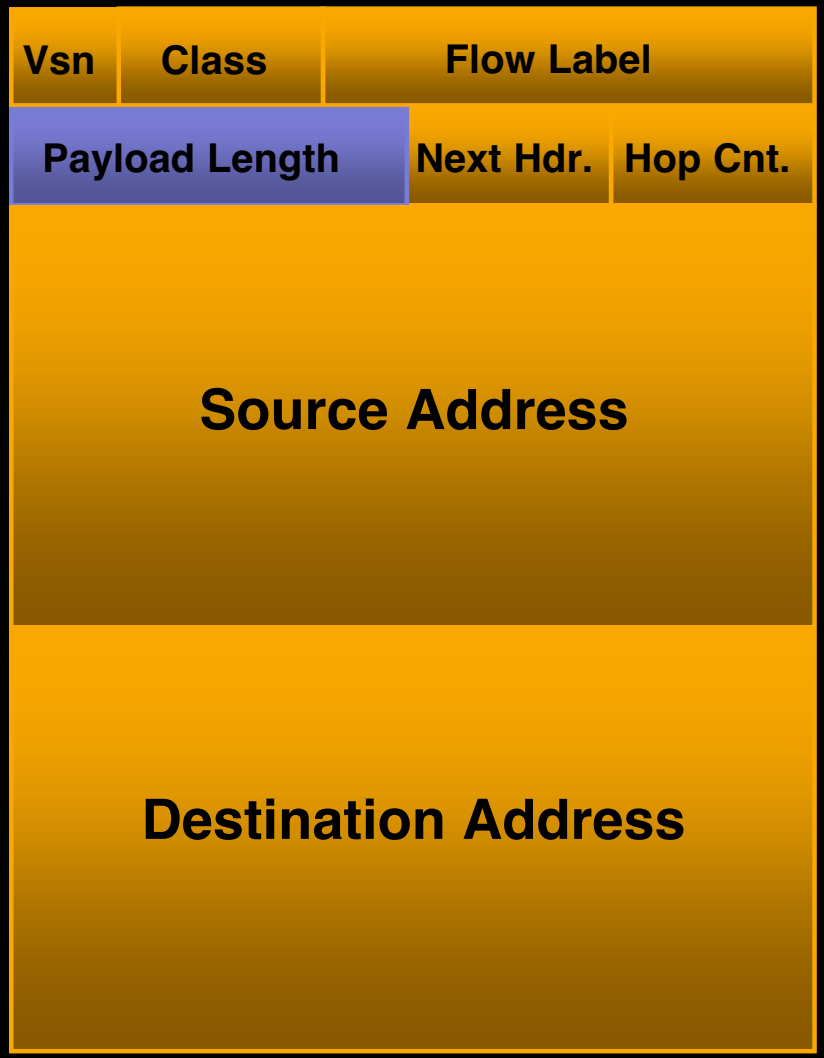
IPv6 Header Structure



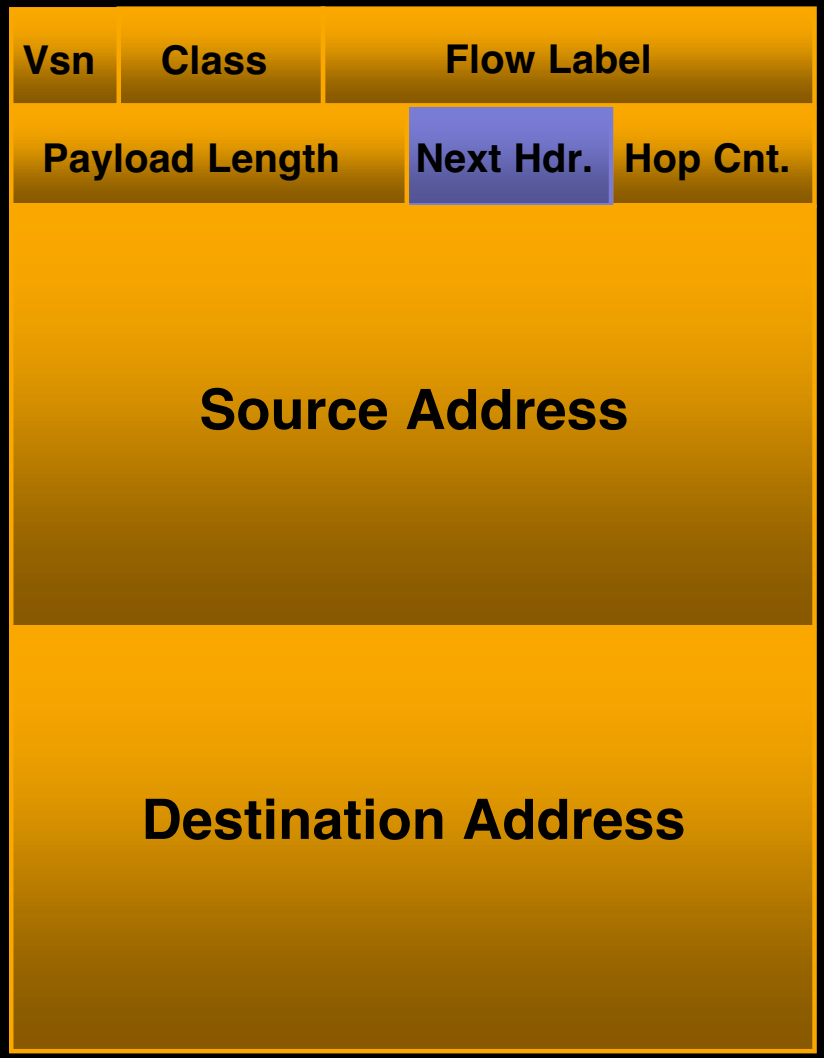
IPv6 Header Structure



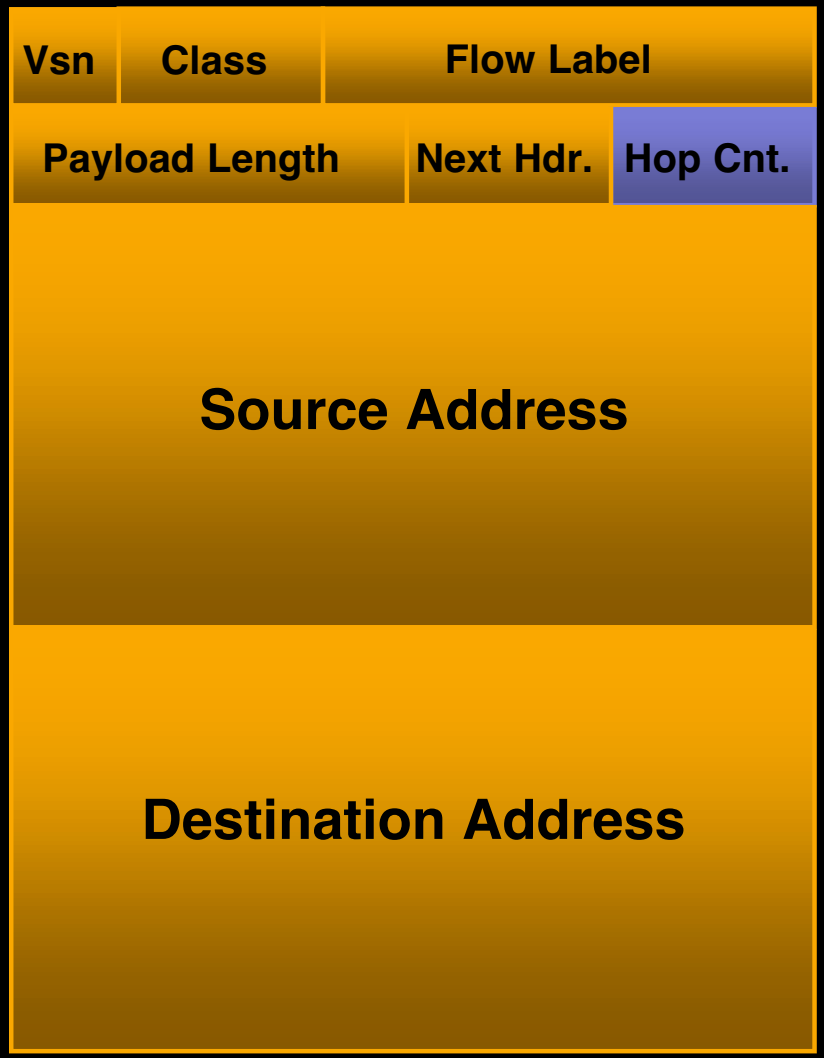
IPv6 Header Structure



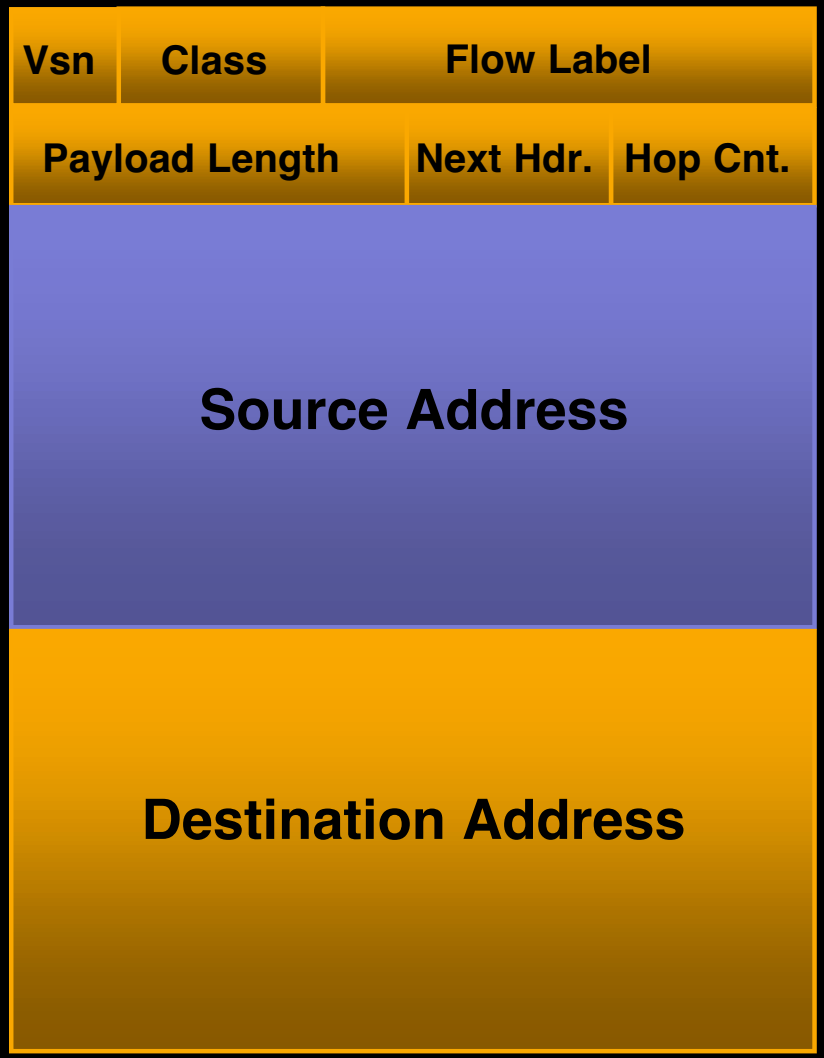
IPv6 Header Structure



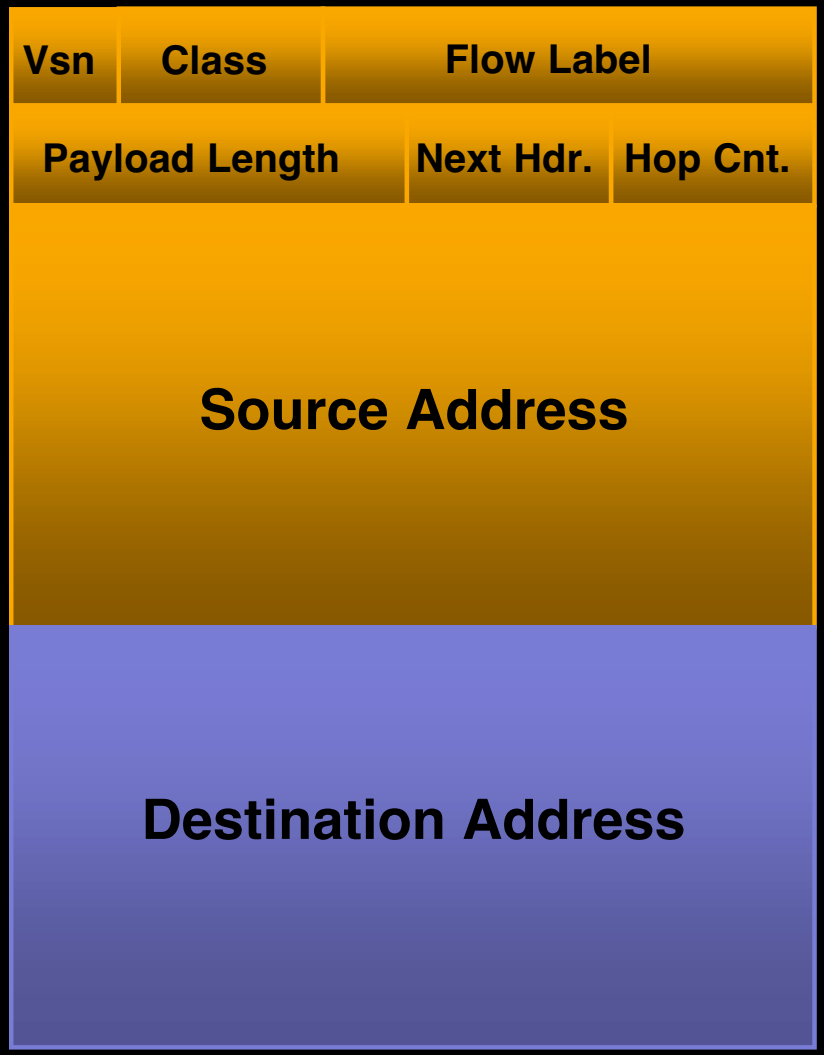
IPv6 Header Structure



IPv6 Header Structure



IPv6 Header Structure



IPv6 Addresses

- 128 bits instead of 32
- From 4 billion to 340 undecillion (3.4 * 10 ^ 38)
- Addressing architecture
    - Classless Interdomain Routing
    - Address Structure
        - Unicast Addresses
        - Local Use Addresses
        - Multicast

IPv6 Address Architecture

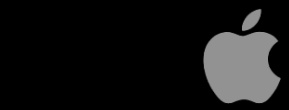
- IPv6 Address Scopes
  - Link-Local
  - Site-Local
  - Global

ICMPv6

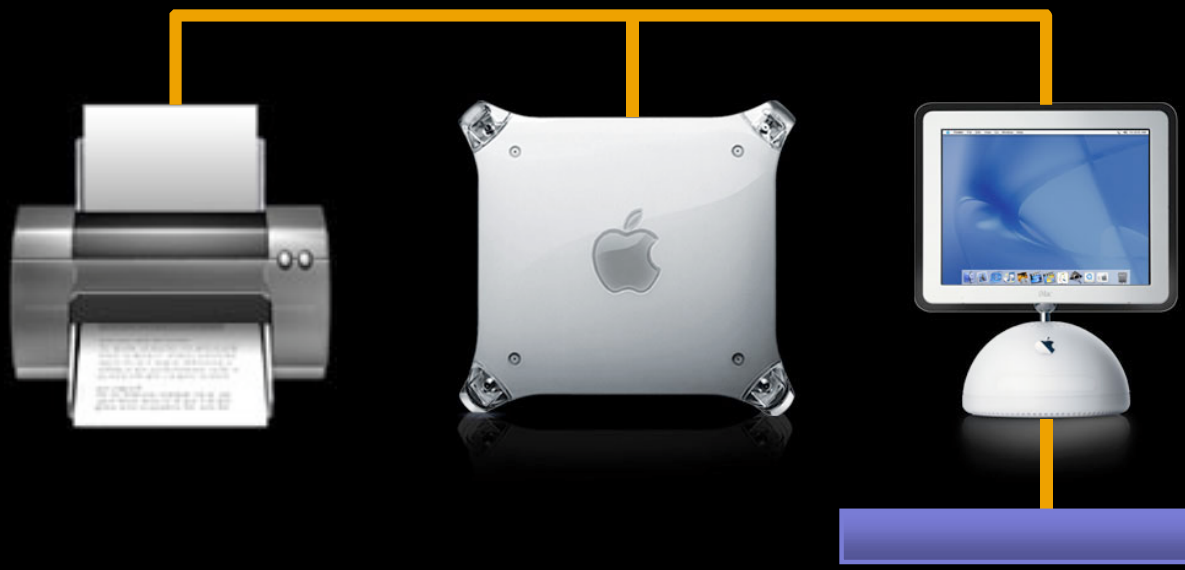
- Error messages
- Echo request/reply
- Multicast Listener Discovery
- Neighbor Discovery
    - Replacement for ARP
    - Router Discovery
    - Prefix Discovery

IPv6 Auto-Configuration

- AppleTalk-like ease of setup in an IP environment
    - Improved support for networking in the home, ad hoc networking, etc.
    - Auto-configuration designed in from the start
- Fits in with the ideas behind Rendezvous

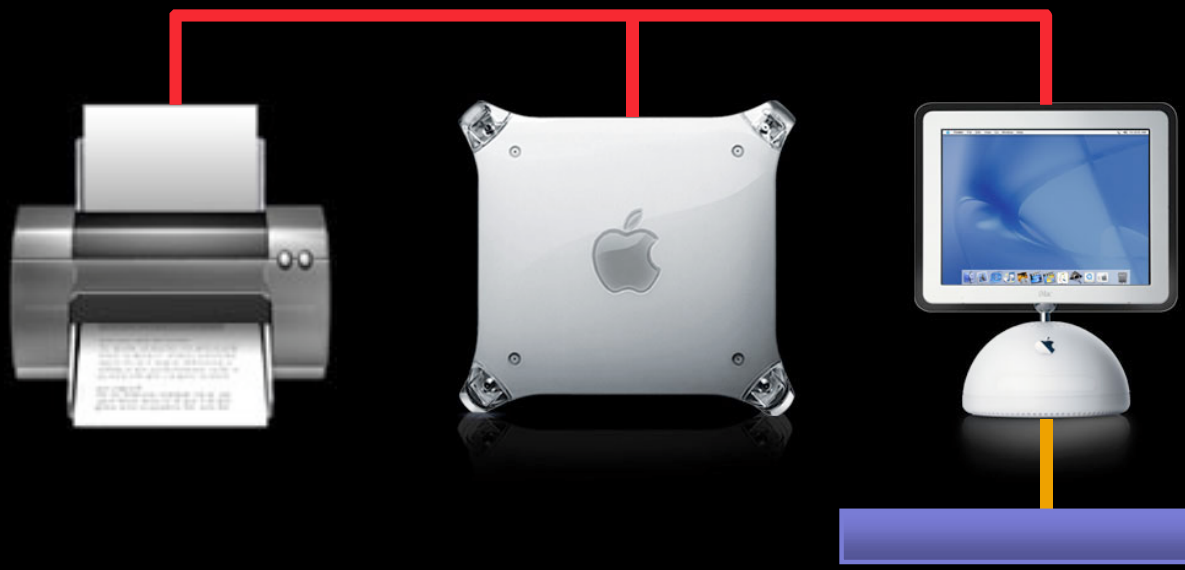


How Auto-Configuration Works



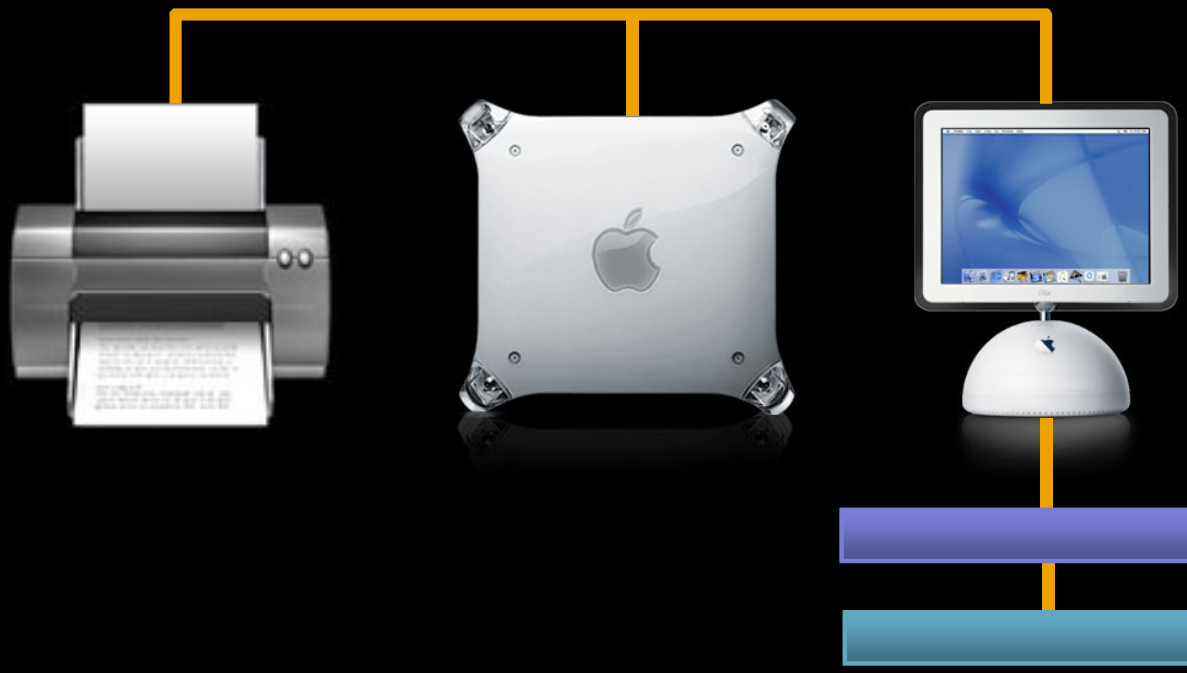
Link Local Address

How Auto-Configuration Works

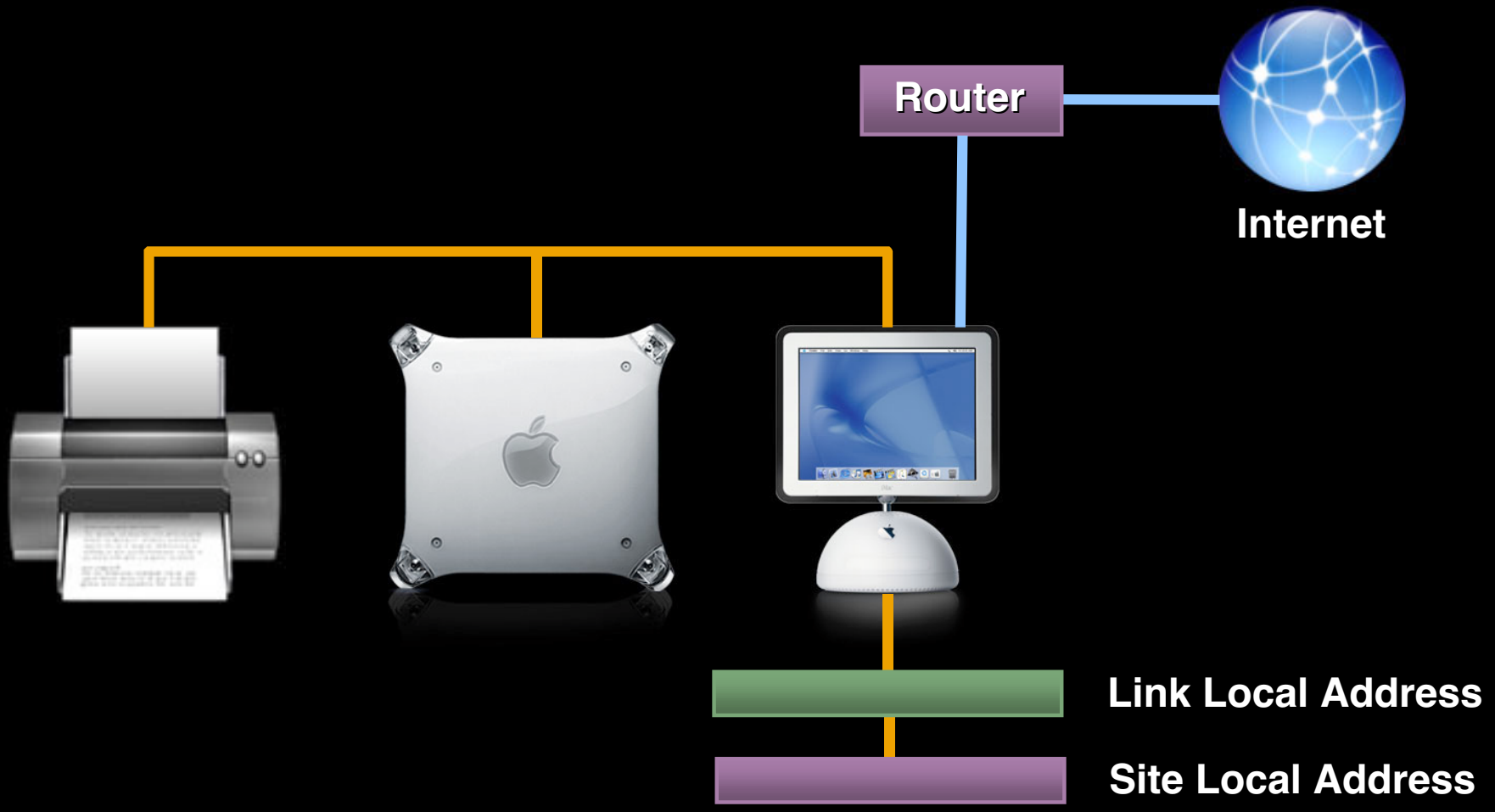


Link Local Address

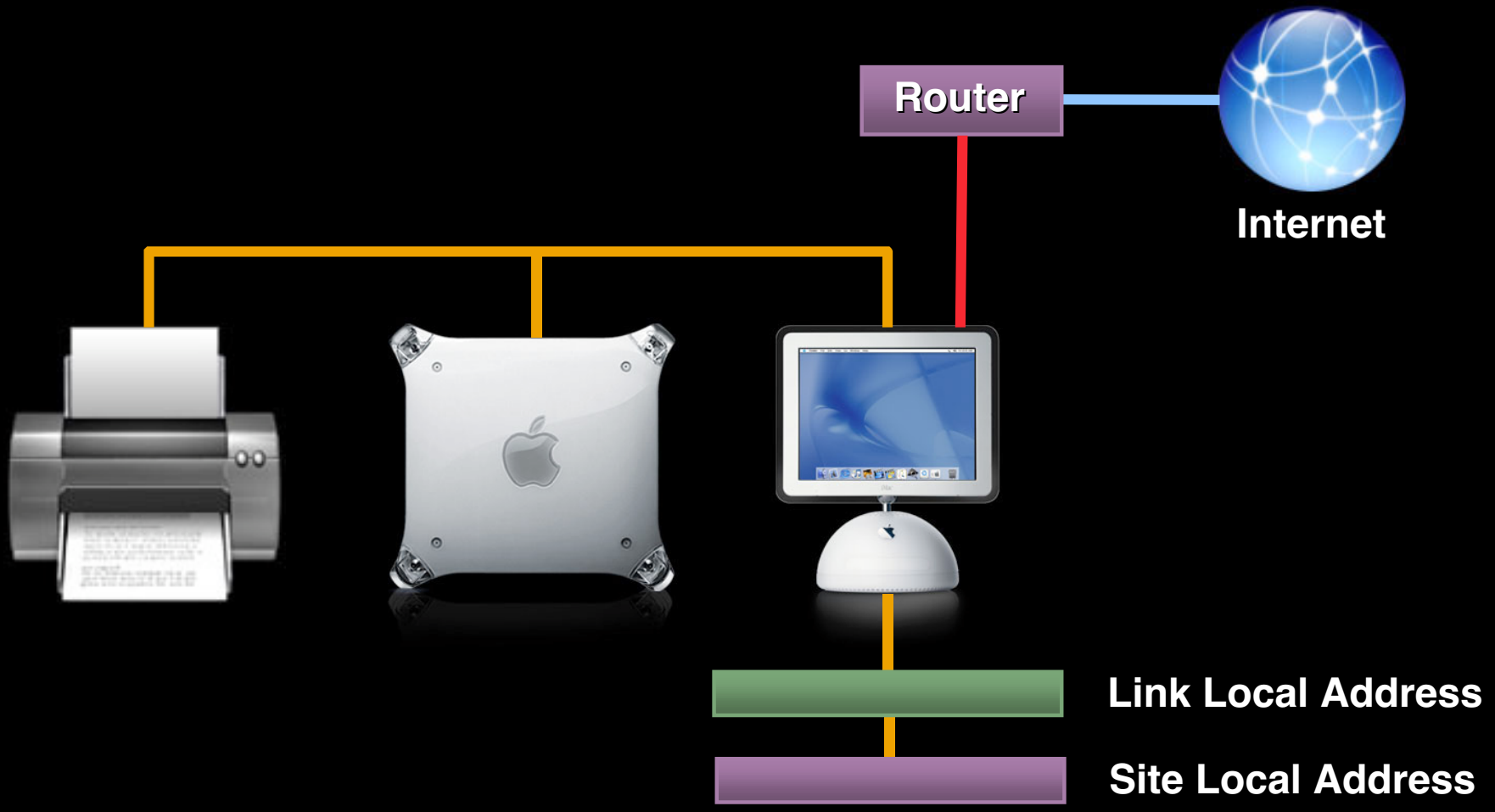
How Auto-Configuration Works



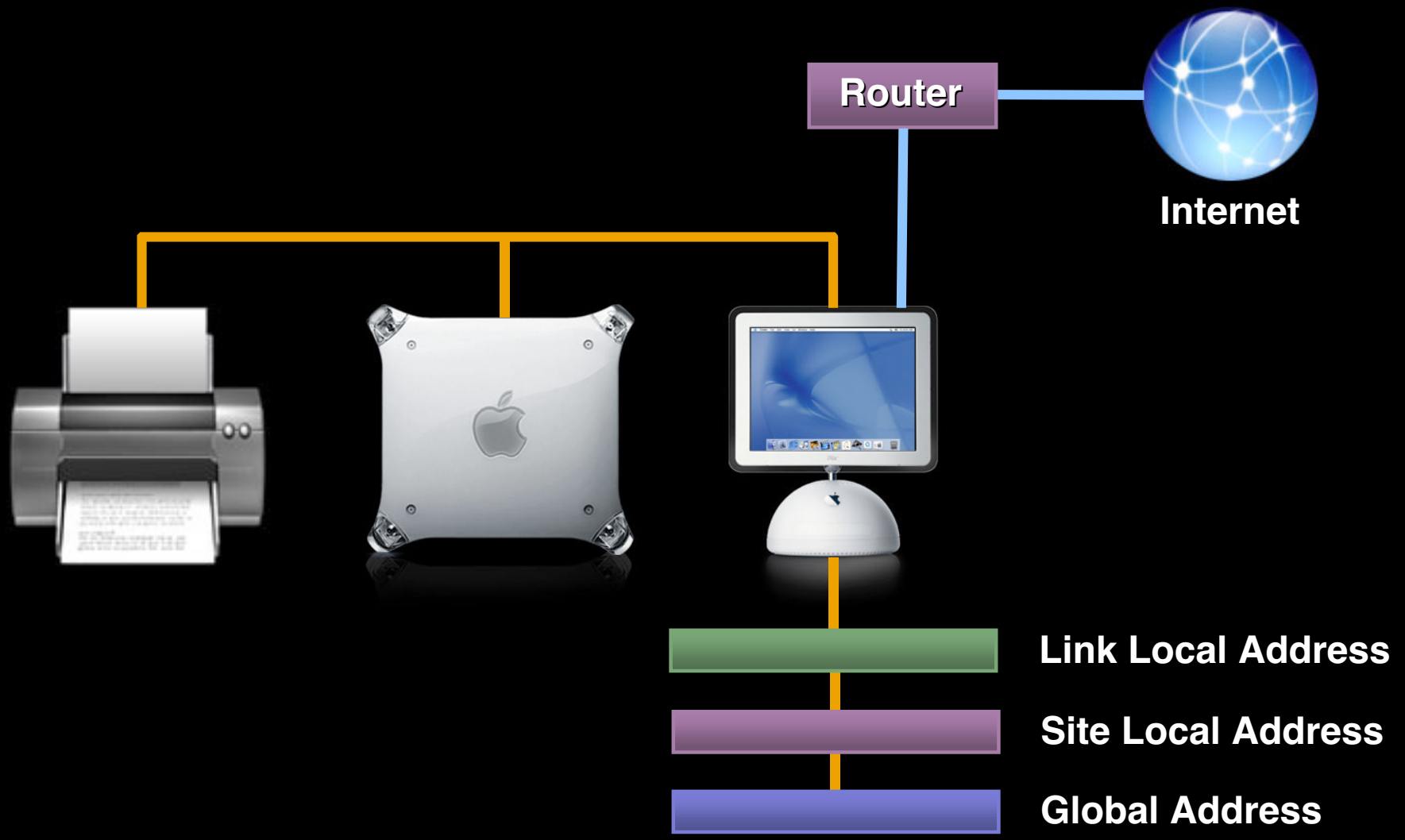
More Auto-Configuration



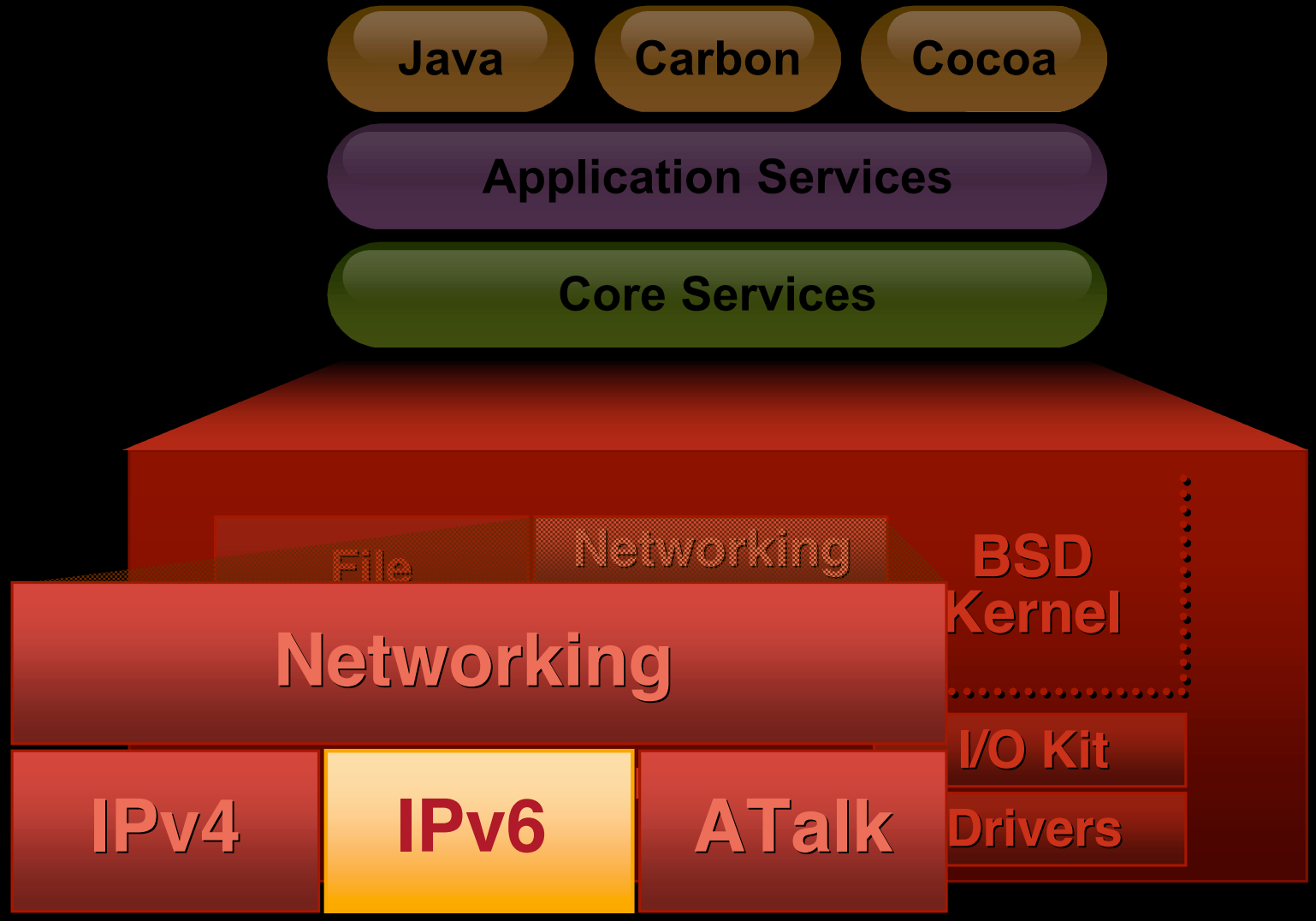
More Auto-Configuration



More Auto-Configuration



IPv6 in Mac OS X

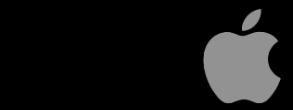


IPv6 in Mac OS X

- Available in Jaguar
  - For developers and advanced users
  - No support in System Configuration yet
- Based on the Open Source work of KAME
  - Standard for BSD implementations
- Kernel, command-line tools, headers and libraries

IPv6 in Mac OS X

- Support for Transition
  - Dual Stack implementation
  - 6to4: IPv6 tunnel over IPv4
  - IPv6 Routing
- Auto-configuration
  - Neighbor Discovery
- DNS for IPv6 in Jaguar
  - AAAA records via IPv4 DNS only



IPv6 in Mac OS X

- Tools
  - ifconfig, ping6, traceroute6, netstat, route, tcpdump, rtsol
- Apps
  - ssh, ftp, telnet
- Good examples of Protocol Independent implementation

Be IP Version Agnostic

- IPv6 addresses are bigger (128 bits)
- Interfaces have multiple addresses
  - Do not identify an interface by an address
  - Do not assume AF_INET length, use **getifaddrs**
- Addresses can be autoconfigured
  - Multiple scopes

IPv6 in Mac OS X: APIs

- Basic BSD APIs (RFC 2553)
  - IPv4 Binary compatibility
  - Allows for IPv6/IPv4 independent programming
- **getaddrinfo** is the main address independent API
  - Use it instead of **gethostbyname**
- Check Jaguar man pages for new resolver calls:
  - **getnameinfo, getipnodebyname, inet_pton, inet_ntop**

IPv6 in Mac OS X: APIs

- Advanced BSD APIs (RFC 2292)
  - Provides access to new IPv6 only functionalities
  - Raw IPv6 socket standardization
- For higher level APIs—use CFNetwork
- For more details about protocol independence
  http://www.kame.net/newsletter/19980604

Roadmap

100 The Darwin Road Map	Room A1 **Mon., 2:00pm**
107 The Darwin Kernel	Civic **Wed., 9:00am**
108 Managing Kernel Extensions	Civic **Wed., 10:30am**
906 Developing for Performance	Hall 2 **Fri., 9:00am**
FF012 Core OS Networking	Room J1 **Fri., 2:00pm**

Who to Contact

Tom Weyer
Network and Communications Evangelist
weyer@apple.com

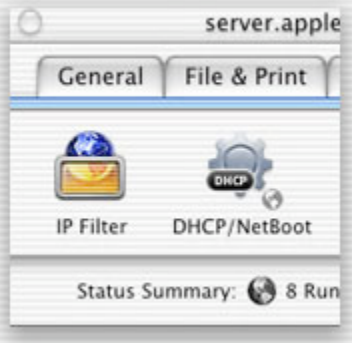
Vincent Lubet
Manager, Core OS Networking Team
vlubet@apple.com

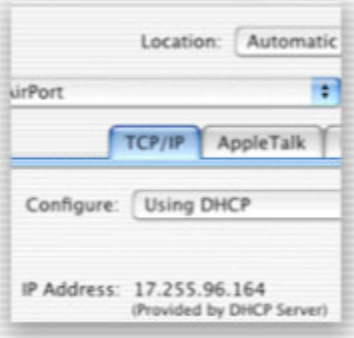
http://developer.apple.com/wwdc2002/urls.html

For More Information

- KAME IPv6 and IPSec
  http://www.kame.net
- FreeBSD How-To
  http://www.freebsd.org/handbook/ipsec.html
- Other performance tips: Mac OS X Perf book
  http://developer.apple.com/techpubs/macosx/Essentials/Performance
- Darwin programming documentation
  http://developer.apple.com/techpubs/macosx/Darwin

Q&A





Tom Weyer
Network and Communications Evangelist
weyer@apple.com

http://developer.apple.com/wwdc2002/urls.html


WWDC2002


WWDC2002


WWDC2002